Vol.2011-HPC-130 No.65
2011/7/29

3次元FDTDカーネルのキャッシュメモリを考慮した性能改善の実装と性能評価

南　武志[†1]　岩下　武史[†1]　中島　浩[†1]

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し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，タイル間での冗長な計算がオーバー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，タイルの位置と形状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．

Cache-Aware Implementation for Performance Improvement of Three-Dimensional FDTD Kernel and its Performance Evaluation

Takeshi Minami,[†1] Takeshi Iwashita[†1]
and Hiroshi Nakashima [†1]

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. By updating electrical and magnetic fields in each small domain 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 between adjacent tiles. 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 and shape 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.

1. はじめに

計算機による電磁場解析シミュレーションは重要な科学技術計算の一つであるが，解析対象の大規模化・複雑化に伴い計算時間が増大しており，シミュレーション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のための手法の一つとして，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一つである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及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，複数のコアが同じメモリ帯域を利用するため，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Iterative stencil computation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

†1 京都大学
Kyoto University

Vol.2011-HPC-130 No.65
2011/7/29

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．著者らは文献 5) 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）手法による場合と比べて性能向上が得られた．一方，文献 5) 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，タイル間で格子点を重複し，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，タイルの形状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と提案する．同手法を一般的な 3 次元 FDTD 法による実装（Naive な実装）と比較し，性能の向上がみられることを示す．

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毎に陽的に解いていく電磁場数値解析手法の一つである．本研究では離散電磁場変数の配置に Yee のメッシュを用い，電磁場計算では Leap-flog アルゴリズムを用いる[3]．

電磁場のふるま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と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り返しにより，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

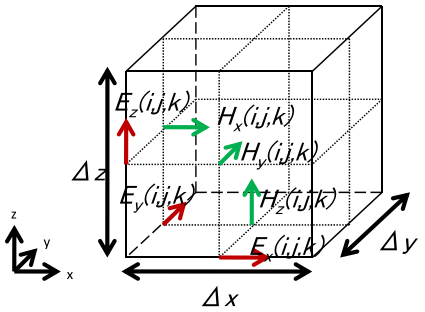

図 1 Yee のメッシュ
Fig. 1 Yee-Cell

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

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

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．

解析空間上の x 軸，y 軸，z 軸での座標を (x,y,z) と表記し，格子座標 (i,j,k) での電場を $\boldsymbol{E}(i,j,k)$，磁場を $\boldsymbol{H}(i,j,k)$ と表す．**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，一般的に以降で Naive な実装法と呼ぶ**図 2** のような方法で実装される．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，スカラ係数 C_e, C_{erx}, C_{ery}，C_{erz}, C_{hrx}, C_{hry}，C_{hrz} を媒質ごとにあらかじめ計算して配列に格納しておき，それを格子点の媒質の種類を定める整数配列 $id(i,j,k)$ を介して参照するものとしている．

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．従って，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．

```
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

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

前節で述べたように，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）を単位として行う事で，配列要素の参照間隔を短くすることで時間的局所性を向上し，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．著者らは参考文献 5) において，複数タイムステップの計算を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，一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

提案手法による 3 次元 FDTD 法の実装では，解析領域をまずタイルと呼ぶ小領域に分割し，各タイル上で複数ステップの電磁場の更新を行う．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全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．

以下に提案手法の詳細について述べる．解析領域全体のサイズを $n_x \times n_y \times n_z$ 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プステップを s_t と表す．ここでは，全ての格子点の電場・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 つの要素をもつ配列 $\boldsymbol{s}_{i,j,k}$，

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

を導入する．

ここで，図 2 のような実装をすると，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，$\boldsymbol{s}_{i,j,k} = (u, u), u \in \{0, \ldots, s_t\}$ を $\boldsymbol{s}_{i,j,k} = (u+1, u)$ に更新するためには $\boldsymbol{s}_{i-1,j,k}, \boldsymbol{s}_{i,j-1,k}, \boldsymbol{s}_{i,j,k-1}$ が (u, u) または $(u+1, u)$ である必要があり，$\boldsymbol{s}_{i,j,k} = (u+1, u)$

を $\boldsymbol{s}_{i,j,k} = (u+1, u+1)$ に更新するためには $\boldsymbol{s}_{i+1,j,k}, \boldsymbol{s}_{i,j+1,k}, \boldsymbol{s}_{i,j,k+1}$ が $(u+1, u)$ 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．

タイル $T_{I,J,K}$ に対して，その位置をずらした領域 $T^u_{I,J,K}$ を以下のように定義する．

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

提案手法では，あるタイルの更新計算中において，時刻 $t+u$ での計算は，(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 $T^{u+1}_{I,J,K}$ の更新を行うこととする．

また，タイルの計算順序は辞書式順序付けによるものとする．

3.2 提案手法が可能であることの証明

以下でこの手法により正しい更新が可能であることを示す．

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 $S_{I,J,K} = \{(L,M,N)|(L < I) \cup (L = I \cap M < J) \cup ((L = I \cap M = J) \cup N < K)$，$(I, J, K)$ よりも後に位置するタイル座標の集合を $N_{I,J,K} = \{(L,M,N)|(L > I) \cup (L = I \cap M > J) \cup ((L = I \cap M = J) \cup N > K)$ とし，

$$S^u_{I,J,K} = \{(i,j,k)|(i,j,k) \in T^u_{L,M,N}, (L,M,N) \in S_{I,J,K}\}, \quad (11)$$

$$N^u_{I,J,K} = \{(i,j,k)|(i,j,k) \in T^u_{L,M,N}, (L,M,N) \in N_{I,J,K}\}, \quad (12)$$

とする．このとき，式 (10) より，

$$T^u_{I,J,K} \cap S^u_{I,J,K} = T^u_{I,J,K} \cap N^u_{I,J,K} = S^u_{I,J,K} \cap N^u_{I,J,K} = \emptyset, \quad (13)$$

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，

$$\begin{aligned} A^u_{I,J,K} \subseteq \{(i,j,k)|\ & \max((I-1)t_x-u, 1) \le i \le \min(I\cdot t_x - u, n_x), \\ & \max((J-1)t_y-u, 1) \le j \le \min(J\cdot t_y - u, n_y), \\ & \max((K-1)t_z-u, 1) \le k \le \min(K\cdot t_z - u, n_z)\}, \end{aligned} \quad (14)$$

となり，式 (10)，式 (13) より，

$$A^u_{I,J,K} \subseteq \bigcup_{L\ge I, M\ge J, N\ge K} T^{u+1}_{L,M,N} \subseteq T^{u+1}_{I,J,K} \cup N^{u+1}_{I,J,K} \subseteq \overline{S^{u+1}_{I,J,K}}, \quad (15)$$

が成り立つ．同様に，

$$A^u_{I,J,K} \subseteq \bigcup_{L\le I, M\le J, N\le K} T^u_{L,M,N} \subseteq T^u_{I,J,K} \cup S^u_{I,J,K} \quad (16)$$

となる．ここで，式 (15)，式 (16) より，

$$A^u_{I,J,K} \subseteq (T^u_{I,J,K} \cup S^u_{I,J,K}) \backslash S^{u+1}_{I,J,K} \subseteq T^u_{I,J,K} \cup (S^u_{I,J,K} \backslash S^{u+1}_{I,J,K}). \quad (17)$$

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，

$$B^u_{I,J,K} \subseteq \bigcup_{L\ge I, M\ge J, N\ge K} T^u_{L,M,N} \subseteq T^u_{I,J,K} \cup N^u_{I,J,K} \subseteq \overline{S^u_{I,J,K}}, \quad (18)$$

$$B^u_{I,J,K} \subseteq \bigcup_{L\le I, M\le J, N\le K} T^{u-1}_{L,M,N} \subseteq T^{u-1}_{I,J,K} \cup S^{u-1}_{I,J,K}, \quad (19)$$

となり，式 (18)，式 (19) より，

$$B^u_{I,J,K} \subseteq (T^{u-1}_{I,J,K} \cup S^{u-1}_{I,J,K}) \backslash S^u_{I,J,K} \subseteq T^{u-1}_{I,J,K} \cup S^{u-1}_{I,J,K} \backslash S^u_{I,J,K}. \quad (20)$$

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，

$$\boldsymbol{s}_{i,j,k} = \begin{cases} (s_t, s_t), & (i,j,k) \in S^{s_t}_{I,J,K} \\ (u+1, u), & (i,j,k) \in S^u_{I,J,K} \setminus S^{u+1}_{I,J,K}, s_t - 1 \ge u \ge v, \\ (v, v), & (i,j,k) \in T^v_{I,J,K} \\ (u+1, u), & (i,j,k) \in (S^u_{I,J,K} \cup T^u_{I,J,K}) \setminus (S^{u+1}_{I,J,K} \cup T^{u+1}_{I,J,K}), \\ & v-1 \ge u \ge 0 \\ (0,0), & (i,j,k) \notin S^0_{I,J,K} \cup T^0_{I,J,K}, \end{cases}, \quad (21)$$

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 (21) は，

$$\boldsymbol{s}_{i,j,k} = (0,0), \forall (i,j,k), \quad (22)$$

となり，明らかに式 (21) 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 (21) 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．この時，式 (17) および式 (21) より，

$$\boldsymbol{s}_{i,j,k} = (v+1, v), \ (i,j,k) \in A^v_{I,J,K} \cap S^v_{I,J,K}, \quad (23)$$

$$\boldsymbol{s}_{i,j,k} = (v, v), \ (i,j,k) \in T^v_{I,J,K}, \quad (24)$$

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．この時式 (20) および電場更新の結果より，

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

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

$$\boldsymbol{s}_{i,j,k} = \begin{cases} (s_t, s_t), & (i,j,k) \in S^{s_t}_{I,J,K} \\ (u+1, u), & (i,j,k) \in S^{u}_{I,J,K} \setminus S^{u+1}_{I,J,K}, s_t - 1 \geq u \geq v+1, \\ (v+1, v+1), & (i,j,k) \in T^{v+1}_{I,J,K} \\ (u+1, u), & (i,j,k) \in (S^{u}_{I,J,K} \cup T^{u}_{I,J,K}) \setminus (S^{u+1}_{I,J,K} \cup T^{u+1}_{I,J,K}), \\ & v \geq u \geq 0 \\ (0,0), & (i,j,k) \notin S^{0}_{I,J,K} \cup T^{0}_{I,J,K}, \end{cases} \tag{26}$$

となり，時刻 $t+v+1$ においても式 (21) 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 (21) は，

$$\boldsymbol{s}_{i,j,k} = \begin{cases} (s_t, s_t), & (i,j,k) \in S^{s_t}_{I,J,K} \cup T^{v}_{I,J,K} \\ (u+1, u), & (i,j,k) \in (S^{u}_{I,J,K} \cup T^{u}_{I,J,K}) \setminus (S^{u+1}_{I,J,K} \cup T^{u+1}_{I,J,K}), \\ & s_t - 1 \geq u \geq 0 \\ (0,0), & (i,j,k) \notin S^{0}_{I,J,K} \cup T^{0}_{I,J,K}, \end{cases} \tag{27}$$

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 (27) は，

$$\boldsymbol{s}_{i,j,k} = \begin{cases} (s_t, s_t), & (i,j,k) \in S^{s_t}_{I',J',K'} \\ (u+1, u), & (i,j,k) \in S^{u}_{I',J',K'} \setminus S^{u+1}_{I',J',K'}, \\ & s_t - 1 \geq u \geq 0 \\ (0,0), & (i,j,k) \notin S^{0}_{I',J',K'}, \end{cases} \tag{28}$$

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 (21) と等しい．よって，帰納的に式 (21) はすべてのタイル，$v = \{0, \cdots s_t\}$ で成り立つ．

式 (21) を用いてすべてのタイルを計算し終えた時点で，

$$\boldsymbol{s}_{i,j,k} = (s_t, s_t), \forall (i,j,k), \tag{29}$$

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 (21) 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 提案手法における電磁場更新の手順

3.2 節での議論を踏まえ，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，$T^{u}_{I,J,K}, u = \{0, \cdots S_t\}$ 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，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．

4. 性能評価

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ノードである富士通 HX600[4] 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 32GB (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 36.8GFlops，コアあたり 9.2GFlops，キャッシュメモリ容量は L2 キャッシュがコアあたり 512KB，L3 キャッシュがプロセッサあたり 2MB 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 10.6GB/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，解析領域中の一方向の格子点数を 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．実応用を考慮して，本論

文で主に対象とするのは一方向の格子点数 n が 200〜250 程度の場合である．

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．スレッド並列処理を行う場合，本実験ではタイルを各タイムステップごとに x 方向にスレッド数で分割するものとする．なお，本実験では各スレッドを一つのプロセッサ（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．

Naive な実装と提案手法の各々を用いた場合について，格子サイズを変化させ，両者の 1 格子点・タイムステップあたりの計算時間を求めて比較する．提案手法を用いた場合では，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，その性能を評価する．1 格子点・タイムステップあたりの計算時間は，Naive な実装と提案手法を用いた場合ともに，一方向の格子サイズ n を 200〜250 の範囲で変化させた場合の平均値とする．

図 3，**図 4**，**図 5** に，逐次実行，2 スレッド実行および 4 スレッド実行における提案手法の測定結果を示す．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 t_t を示している．比較のため，同じ並列度で実行した Naive な実装の計算時間もグラフ内に示す．

これらのグラフより，提案手法は適切なタイルサイズ n_t とステップ数 s_t を用いた時，Naive な実装を上回る性能を発揮していることが見て取れる．一方で，タイルサイズ n_t が最適値よりも大きい場合，タイルの持つデータ量が十分にキャッシュメモリ上で計算できる値を上回ることにより，キャッシュメモリの活用効果がタイルサイズの増加に伴い低下し，計算時間が増加する．逆にタイルサイズ n_t が最適なサイズと比較して小さい場合にも計算時間は増加する．この時，タイルはキャッシュ上に収納可能であると考えられるが，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の連続性が下がることによるによるオーバーヘッドが存在することなどが計算時間の増加の原因であると考えられる．

また，スレッド数の増加とともに提案手法の Naive な手法に対する相対的な有効性は高まる傾向にある．これは，スレッド数の増加に伴いスレッドあたりのメモリ帯域が狭まったため，キャッシュの利用による効果の影響が高まったためと考えられる．図 5 より，提案手法による実装は 4 スレッド並列で実行された場合，Naive な実装と比べ，実行時間が最大で 52%短縮されることが確認できる．

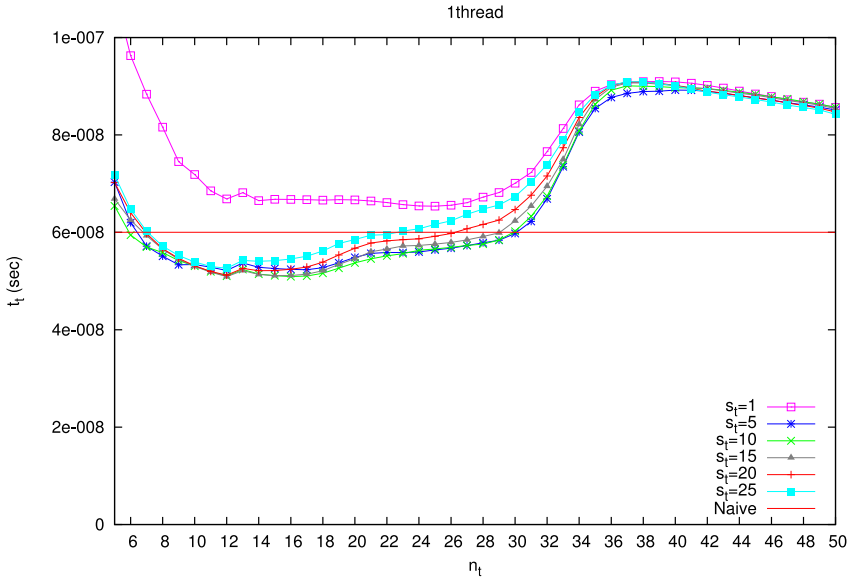

図 3 逐次実行における性能
Fig. 3 Performance (serial)

図 6 は各並列数にいて最適なタイリングのステップ数 s_t を用いた場合の計算時間を比較したものである．横軸がタイリングのステップ数 s_t，縦軸が 1 格子点・タイムステップあたりの計算時間を示している．また，タイルサイズ n_t はそれぞれの並列度においてもっとも計算時間の短くなる n_t とする．逐次実行の場合は $n_t = 16$，2 スレッド並列の場合は $n_t = 21$，4 スレッド並列の場合は $n_t = 32$ である．また，比較のため，同じ並列度で実行した Naive な実装の計算時間および Naive な手法のキャッシュに収納可能なサイズの場合の計算時間もグラフ内に示す．

提案手法ではタイリングのステップ数 s_t を変化させてもある一定以上の範囲では 1 タイムステップ・格子点あたりの計算時間は変化しない．これは，s_t が大きいとタイル $T_{I,J,K}$ 内での計算のキャッシュ利用率は上がるが，逆にタイル $T_{I,J,K}$ の更新終了後に隣接するタイル $T_{I',J',K'}$ の計算に使用するタイル $T_{I,J,K}$ 中に存在していたデータがキャッシュ上に残らなくなるため，この時メインメモリにアクセスする必要があり，結果的にタイリングの効

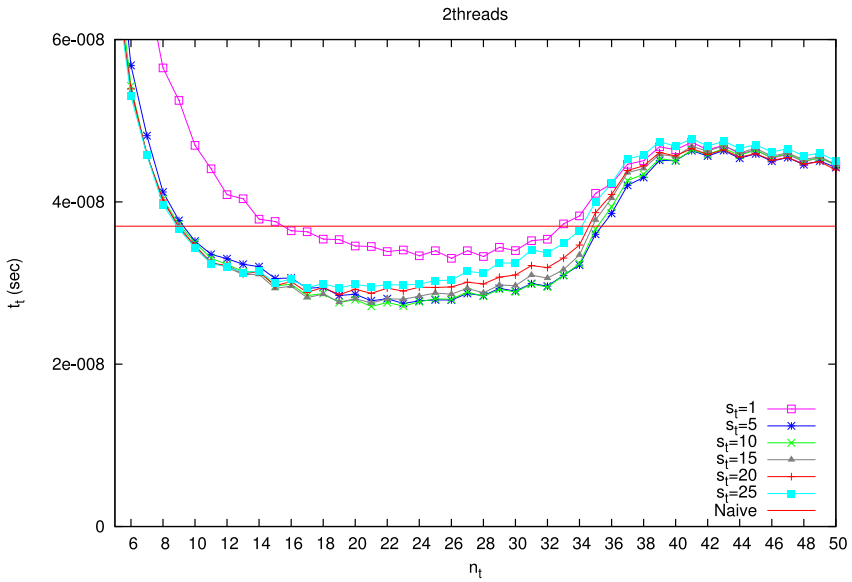

図 4 2 スレッド並列実行における性能
Fig. 4 Performance (2 threads)

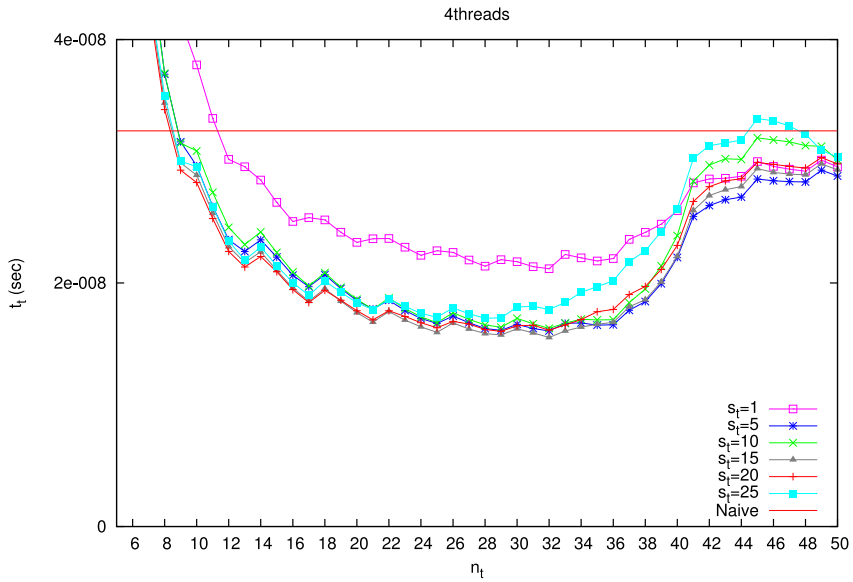

図 5 4 スレッド並列実行における性能
Fig. 5 Performance (4 threads)

果を阻害するためである．

Naive な手法において解析領域がキャッシュ内におさまらない場合，収まる場合と比較するとスレッド数が増えるに従い計算時間の比率は逐次実行の場合 2.16 倍，2 スレッド実行の場合 2.36 倍，4 スレッド実行の場合 3.25 倍と増加する．一方で，提案手法の性能を，Naive な手法をキャッシュ上で計算した場合と比較すると，その計算時間は逐次実行の場合 1.83 倍，2 スレッド実行の場合 1.73 倍，4 スレッド実行の場合 1.55 倍であり，スレッド数が増えるに従いキャッシュ上での計算時間に近づいていることがわかる．これは，Naive な手法ではスレッド数の増加に伴うスレッドあたりのメモリ帯域の減少が大きな影響を及ぼしているのに対して，提案手法を用いた場合はスレッド数が増加することでより大きなタイルをキャッシュ上で計算可能で，キャッシュの利用率が高まり，メモリへのアクセス量が減少しメモリの帯域の影響が減少するためと考えられる．

5. ま と め

本研究では，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，キャッシュメモリの有効性を改善し計算性能を向上させる方法を提案した．

3 次元 FDTD 法の計算性能はプロセッサの演算性能よりもメモリアクセス性能に影響されやすいことに着目し，キャッシュメモリのヒット率を向上し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タイリングが存在する．タイリングは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を個別に行う手法であるが，この手法を 3 次元 FDTD 法に対して適用した場合，隣接するタイル間の影響により，各タイムステップで固定のタイルを用いる場合には，冗長な計算が必要である．

そこで本論文ではタイルの位置と形状を時間ステップごとに変化させ，タイルの計算順序

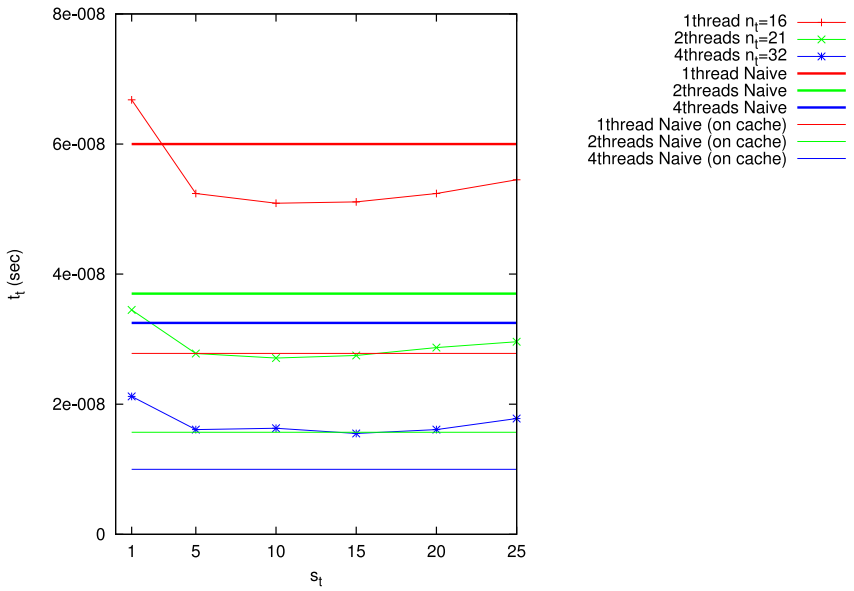

図 6 適切なタイルサイズでの提案手法の性能
Fig. 6 Performance on the appropriate tile size

を適切に設定することで計算量の増加を防ぐ手法を提案した．提案手法により，Naive な実装と同じ計算量でタイリングによるキャッシュメモリのヒット率の向上を得ることが可能となる．提案手法の評価は，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ノードである富士通 HX600 で，AMD 社製クアッドコア Opteron(8356) プロセッサを用いて実施した．数値実験の結果，提案手法は適切なタイムステップ数やタイルサイズを設定することで，Naive な実装法に対して性能の向上を得た．また，並列度の上昇とともに提案手法の Naive な実装に対する相対的な性能は上昇することを確認した．これはスレッド数の増加によりメインメモリの帯域に対する負荷が増大するため，タイリングによるキャッシュメモリのヒット率向上による影響がより顕著に出たためと考えられる．その結果，4 スレッド並列実行の場合，提案手法は Naive な実装と比べて最大で約 52%の実行時間の短縮を得た．

参 考 文 献

1) K. S. Yee, “Numerical solution of initial boundary value problems involving Maxwell’s equations in isotropic media,” IEEE Trans. Antennas Propagat., vol. AP-14, pp. 302-3-7, Aug. 1966.
2) M. Wolf, “More iteration space tiling,” In Proceedings of Supercomputing ’89, 1989.
3) 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).
4) H. Nakashima, : T2K Open Supercomputer: Inter University and Inter-Disciplinary: Collaboration on the New Generation Supercomputer, in Proc. Intl. Conf. Informatics Education and Research for Knowledge-Circulating Society, pp.137-142, (2008).
5) 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, 70-83 (Mar. 2011).